01 - Introduction to Computing using R

Junvie Pailden

SIUE, F2017, Stat 589

August 22, 2017

Why R

- `R` offers a powerful and appealing interactive environment for exploring data, running simulations, etc.
- `R` is platform independent meaning it is available on Windows, Mac, and Linux.
- `R" has excellent resources both online (just google any issue/question) and using help(...), e.g.`help(lm)‘.
- `R` is not a black box software, i.e., you can trace how a function or package works by following the `{R} script, e.g. lm()`

R as a calculator

```
# create an integer sequence
3:7
```

```
# [1] 3 4 5 6 7
```

```
# create an sequence from 0 to 3 with 0.5 increment
seq(0,3,by=0.5)
```

```
# [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0
```

```
# create a repeated sequence
rep(pi,4)
```

```
# [1] 3.141593 3.141593 3.141593 3.141593
```

```
# Basic Operations
(2-3)*(3/4)
```

Concatenate operator

```
c(6,20,-3) # numbers
```

```
# [1]  6 20 -3
```

```
c(log(100), log10(100))
```

```
# [1] 4.60517 2.00000
```

```
c("words","are","wind") # strings
```

```
# [1] "words" "are"   "wind"
```

Variable assignment, common operator

```
a <- 3
# is the same as
a -> x
# and x = 3
b <- 1:3; c <- 2:4
a + b # scalar multiplication
```

```
# [1] 4 5 6
```

```
b * c # entrywise multiplication
```

```
# [1]  2  6 12
```

Matrices

```
# store the vector (1, 2, 3, 4, 5, 6, 7, 8)
# into a 2x4 matrix named A
# nrow = # of rows, ncol = # of columns
A <- matrix(c(1,2,3,4,5,6,7,8), nrow = 2, ncol = 4)
colnames(A) <- c('C1', 'C2', 'C3', 'C4')
A
```

```
#      C1 C2 C3 C4
# [1,]  1  3  5  7
# [2,]  2  4  6  8
```

```
B <- matrix(1:8, nrow = 2, ncol = 4, byrow = TRUE) # 1:8 is a
B
```

```
#      [,1] [,2] [,3] [,4]
# [1,]    1    2    3    4
# [2,]    5    6    7    8
```

Extracting matrix entries

```
A[1, 3]   # row 1, col 3 entry
```

```
# C3
#  5
```

```
A[1, 1:3] # row 1, col 1 to 3
```

```
# C1 C2 C3
#  1  3  5
```

```
A[1, ]    # row 1
```

```
# C1 C2 C3 C4
#  1  3  5  7
```

```
A[ , 3]   # col 3
```

Data Frames

Data frames are similar in concept to Excel spreadsheet where each column represents a variable or measurement and each row represents and observational unit.

We can check the structure of the data frame using the function `str()`. The output includes data information such as

- dimension (number of observations and variables),
- variable names
- data type
  - `logi`: logical, `TRUE` or `FALSE`
  - `int`: integer
  - `num`: numerical
  - `chr`: character
  - `Factor`: Factors are how R keeps track of categorical variables.

Use `data.frame()`:

```
stark.kids <- data.frame(
  Name = c("Jon", "Sansa", "Arya", "Bran"),
  Age = c(24, 20, 18, 17)
)
str(stark.kids)
```

```
# 'data.frame': 4 obs. of  2 variables:
#  $ Name: Factor w/ 4 levels "Arya","Bran",..: 3 4 1 2
#  $ Age : num  24 20 18 17
```

```
stark.kids
```

```
#    Name Age
# 1   Jon  24
# 2 Sansa  20
# 3  Arya  18
# 4  Bran  17
```

Variable names

We can also use the names of variables in a data frame to access the variable using the notation `data$name`.

```
stark.kids$Name
```

```
# [1] Jon   Sansa Arya  Bran
# Levels: Arya Bran Jon Sansa
```

```
mean(stark.kids$Age) # average age
```

```
# [1] 19.75
```

```
head(stark.kids, 2) # display first 2 rows
```

```
#    Name Age
# 1   Jon  24
# 2 Sansa  20
```

Reading data in R: Example 1.6 Lizard Size Data

Data analysis using R often involves importing or reading data at some point.

```
# File "T1-3.dat"" should be in your working directory
# Specify the working directory
# Session > Set Working Directory > Choose Directory
lizard <- read.table("T1-3.dat")
# assign names to the column variables
colnames(lizard) <- c("mass", "svl", "hls")
head(lizard, 3) # display first 3 rows
```

```
#     mass svl   hls
# 1  5.526  59 113.5
# 2 10.401  75 142.0
# 3  9.213  69 124.0
```

Reading CSV data in R

`Goind Wireless` data reported the estimated percentage of house- holds with only wireless phone service (no land line) for the 50 U.S. states and the District of Columbia.

The URL of the data set `Going Wireless` that we need to read the data is (`https://goo.gl/72BKSf`).

```
wireless.data <- read.csv("https://goo.gl/72BKSf",
                          header = TRUE)
str(wireless.data) # check structure
```

```
# 'data.frame': 51 obs. of  3 variables:
#  $ Wireless: num  13.9 11.7 18.9 22.6 9 16.7 5.6 5.7 20 16.8
#  $ Region  : Factor w/ 3 levels "E","M","W": 2 3 3 2 3 3 1 1
#  $ State   : Factor w/ 51 levels "AK","AL","AR",..: 2 1 4 3
```

Simple data summaries

```
height <- 58:72
weight <- c(115,117,120,123,126,129,132,135,139,
                142,146,150,154,159,164)
mean(height) # mean of height
```

```
# [1] 65
```

```
var(height)  # variance of height
```

```
# [1] 20
```

```
length(height)
```

```
# [1] 15
```

```
cor(height, weight) # correlation
```

Writing Functions in R

```
function.name <- function(arglist){
    expr
}
```

```
my_fun <- function( x, y ){
      x + y
}
my_fun(1,2)
```

```
# [1] 3
```

```
my_fun2 <- function(x, y) x*y
my_fun2(10, 23)
```

```
# [1] 230
```

If the body of the function is only one line, then braces aren't necessary.

apply() function

Applies a function to sections of an array (or matrix) and returns the results in an array (or matrix).

```
# create a matrix of twenty rows, two columns
dat <- cbind(rbinom(n = 20, size = 5, prob = 0.3),
             rnorm(n = 20))
str(dat)
```

```
#  num [1:20, 1:2] 2 1 1 3 3 0 0 4 0 1 ...
```

```
apply(dat, MARGIN = 2, var) # apply variance to columns
```

```
# [1] 1.3052632 0.6889165
```

Conditionals

```
if (arglist is satisfied) {
    do this
} else {
    do this instead
}
```

```
my_cond <- function( x ){
  # function that tells whether a value is > 20.
  if (x > 20) {
      print("x is greater than 20")
  } else {
      print("x is less than 20")
  }
}
my_cond(10)
```

```
# [1] "x is less than 20"
```

Loops in R

For loops iterate through each item in a vector or a list:

```
for (x in vector) {
  do this
}
```

The colon creates a vector, passing each integer from 0 to 4 to the loop.

```
for (x in 0:4) print(x)
```

```
# [1] 0
# [1] 1
# [1] 2
# [1] 3
# [1] 4
```

Two more loops in R are `repeat()` and `while()`. Search this!

Packages in R! Treasure troves of goodies

- An `R` package is a set of related functions and help files, bundled together.
- Normally, all functions within a single package are related: for example, the `stats` package contains functions for statistical analysis.
- There are few public repositories of packages: the largest is CRAN hosted by the `R` foundation with more than 4000 freely packages.
- To use a package, you first need to install it into `R`.
- Among other ways, you can also install `R` packages directly through `R` console using `install.packages()`.
- To load up an `R` package, use the `library()`.

Learn More

1. An Introduction to R by WN Venables
2. Using R Markdown for Class Reports
3. Quick R website
4. Online learning by RStudio
5. R for Data Science by H Wickham
6. Just google it!